COS402- Artificial Intelligence
Fall 2015

Lecture 7: Practical Methods of Solving CNF

Outline

- **Faster inference in special cases**
  - **Forward chaining**
  - **Backward chaining**
- **Algorithms based-on model checking**
  - **DPLL**
  - **WALKSAT**

Some points

- **A Horn clause has at most one positive literal.**
- **A definite clause has exactly one positive literal.**
- **DPLL does recursive exhaustive search of all models for the given CNF.**
- **WALKSAT uses random and greedy search to find a model that may satisfy the given CNF.**

Forward chaining

- **Initially set all symbols false**
- **Start with symbols that are true in KB**
- **When all premises of a horn clause are true, make its head true.**
- **Repeat until you can’t do more.**

Forward chaining (example)

- **KB:**
  - **$p \Rightarrow Q$**
  - **$(L \wedge M) \Rightarrow P$**
  - **$(B \wedge L) \Rightarrow M$**
  - **$(A \wedge P) \Rightarrow L$**
  - **$(A \wedge B) \Rightarrow L$**
  - **B**
  - **A**
- **α: Q**

Backward chaining

- **Start at goal and work backwards**
- **Takes linear time.**

DPLL

- **Do recursive exhaustive search of all models**
- **Set P_1 = T**
- **Recursively try all settings of remaining symbols.**
- **If no model found**
  - **Set P_1 = F**
  - **Recursively try all settings of remaining symbols**

Additional tricks for DPLL

- **Early termination**
- **Pure symbols**
- **Unit clauses**
- **Component analysis**
- **And more …**

WALKSAT

- **Set all symbols to T/F randomly**
- **Repeat MAX times**
  - **If all clauses are satisfied, then return model**
  - **Choose an unsatisfied clause randomly**
  - **Flip a coin**
    - **If head**
      - **flip a symbol in the clause that maximizes # if satisfied clauses**
    - **Else**
      - **flip a symbol selected randomly from the clause.**

Try DPLL and WALKSAT on example CNF

- $(P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2) \wedge (\neg P_1 \vee \neg P_2) \wedge (\neg P_3) \wedge (P_3 \vee \neg P_4 \vee P_5) \wedge (\neg P_4 \vee \neg P_5)$

Review questions: true or false

1. **Forward chaining takes linear time O(n), while n is the number of clauses.**
2. **Backward chaining starts at goal and works backwards. It can be faster than forward chaining because it doesn't waste time on clauses irrelevant to goal.**
3. **Forward chaining can be used on any KB and α.**
4. **Backward chaining can be used on any KB and α.**

Review questions: true or false(con'd)

5. **Forward chaining is basically a resolution algorithm working on definite or horn clauses.**
6. **DPLL is both sound and complete.**
7. **DPLL takes exponential time in worst case.**
8. **WALKSAT is both sound and complete.**
9. **WALKSAT takes exponential time in worst case.**

Announcement & Reminder

- **P1 is due on Tuesday Oct. 13th. (next week!)**

  --- due by midnight, upload your files to CS dropbox

- **W2 has been released and is due on Tuesday Oct. 20th**

  --- Due in class, hard copies.